Accelerating lemma learning using joins
LPAR 2008 – Doha, Qatar

Nikolaj Bjørner, Leonardo de Moura
Microsoft Research
Bruno Dutertre
SRI International
Satisfiability Modulo Theories (SMT)

SMT = SAT + Theories

- Arithmetic
- Bit-vectors
- Arrays
- ...

Accelerating lemma learning using joins
Satisfiability Modulo Theories (SMT)

\[ x + 2 = y \implies f\left(\text{read}\left(\text{write}\left(a, x, 3\right), y - 2\right)\right) = f\left(y - x + 1\right) \]
Satisfiability Modulo Theories (SMT)

\[ x + 2 = y \Rightarrow f(\text{read}(\text{write}(a, x, 3), y - 2)) = f(y - x + 1) \]

Array Theory
Satisfiability Modulo Theories (SMT)

\[ x + 2 = y \implies f(read(write(a, x, 3), y - 2)) = f(y - x + 1) \]
SMT: Some Applications @ Microsoft

Accelerating lemma learning using joins
Z3 is a new solver developed at Microsoft Research.
Development/Research driven by internal customers.
Free for academic research.
Interfaces:

http://research.microsoft.com/projects/z3
SMT = DPLL + Theories

$\neg (a=b \lor f(a)=f(b), \ a < 5 \lor a > 10, \ a > 6 \lor b = 2$}

- Guessing (case-splitting)
- Deducing (BCP + Theory propagation)
- Conflict resolution $\Rightarrow$ Backtracking + Lemma

Most SMT solvers use only the literals from the given formula!
a[0] = 0
if (c_1) { a[1] = 0; } else { a[1] = 1; }
...
if (c_n) { a[n] = 0; } else { a[n] = 1; }
assert(a[0] == 0);
a_1 = write(a_0, 0, 0)
(¬c_1 ∨ a_2 = write(a_1,1,0))
(c_1 ∨ a_2 = write(a_1,1,1))
...
(¬c_n ∨ a_{n+1} = write(a_n,n,0))
(c_n ∨ a_{n+1} = write(a_n,n,1))
read(a_{n+1},0) ≠ 0

It takes O(2^n) time if lemmas do not use new literals!
"Diamonds are eternal"

\[ a_1 \not\sim a_{50} \land \bigwedge_{i=1}^{49} \left[ (a_i \sim b_i \land b_i \sim a_{i+1}) \lor (a_i \sim c_i \land c_i \sim a_{i+1}) \right] \]

Diagram:

- \(a_1\) is connected to \(b_1\) and \(c_1\).
- \(a_2\) is connected to \(b_2\) and \(c_2\).
- \(a_3\) is connected to \(b_3\) and \(c_3\).
- \(a_4\) is connected to \(b_4\) and \(c_3\).
- \(a_{49}\) is connected to \(b_{49}\) and \(c_{49}\).
- \(a_{50}\) is connected to \(b_{50}\) and \(c_{49}\).
SP(E) calculus

It can solve “diamonds” in polynomial time.

The $SP(E)$ calculus

Very slow in practice!
New literals can be created
- Case-splitting (guessing)
- Lemma Learning

Any SP(E) inference can be simulated by DPLL(E+Δ)
How do we create $\Delta$?
Look ahead

Accelerating lemma learning using joins

New literal

S

p

¬p

l

l
Look ahead

Accelerating lemma learning using joins
Define language \( L \) (of new literals). Examples:

- (Bounds) \( x > 5 \)
- (Equality) \( x = y \)
- (Difference) \( x - y < 3 \)

Theory propagation for \( L \)

Join operator for \( L \)
Join: Examples (Bounds)

\[-p \lor q, \quad \neg q \lor x>5, \quad p \lor x>y, \quad y > 4\]
Join: Examples (Bounds)

\( \neg p \lor q, \quad \neg q \lor x > 5, \quad p \lor x > y, \quad y > 4 \)

{ \{ x > 5 \} }
Join: Examples (Bounds)

\[-p \lor q, \quad \neg q \lor x>5, \quad p \lor x>y, \quad y > 4\]
Join: Examples (Bounds)

\[ \neg p \lor q, \quad \neg q \lor x > 5, \quad p \lor x > y, \quad y > 4 \]

\[ \{ x > 5 \} \sqcup \{ x > 4 \} = \{ x > 4 \} \]
Join: Examples (Equalities)

\[
\{ x = y, y = z, x = z \} \cup \{ x = z, z = w, x = w \} = \{ x = z \}
\]
Join: Examples (Difference constraints)

\{ x - y < 3 \} \uplus \{ x - y < 2, y - z < 1, x - z < 3 \} = \{ x - y < 3 \}
Other examples:

- Linear arithmetic: polyhedral.
- Array partial equalities:
  \[ a =_i b \quad (\forall x: x = i \lor a[x] = b[x]) \]
- k-look ahead.
SMT solvers are fast, but they may choke in simple formulas.

DPLL(join) = SMT + “Abstract Interpretation”.

Future work: new literals during conflict resolution.

http://research.microsoft.com/projects/z3